UANL
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FIME
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

Sistemas Operativos

Actividad Fundamental #2

Multitarea y Control de Concurrencia

Investigación

Docente: **Dra. Norma Edith Marín Martínez**

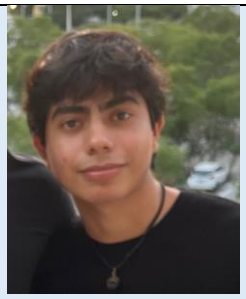
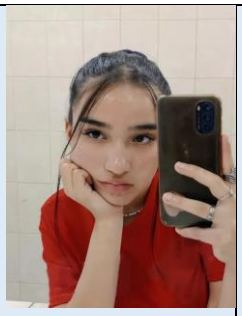
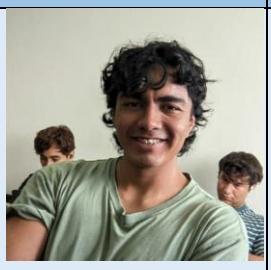
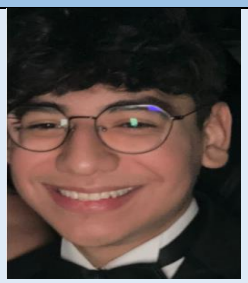
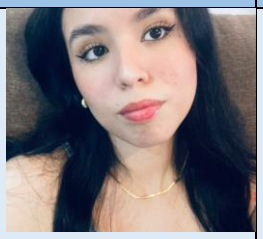
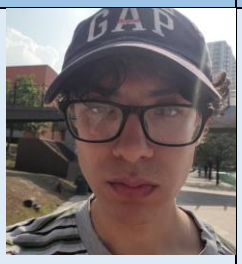
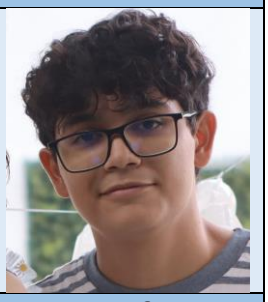
Hora: **M3** Salón: **9104**

Agosto – Diciembre 2024

Matrícula	Alumno	Carrera
1952809	Castillo Arreola Antonio	IAS
2041139	Álvarez García Angel Ricardo	ITS
2044753	Ramírez Núñez Brian Orlando	IAS
2049875	Sainz Coronado Alfonso	ITS
2049903	Almaguer Espinosa Isac Alfredo	IAS
2051321	Garza Walle Alejandra	ITS
2055826	Salinas Monsiváis Emiliano	ITS
2109508	Martínez Tamez Valeria Guadalupe	IAS
2132048	Naranjo Rojas Horacio	ITS
2132062	Loredo Pérez Axel Arturo	ITS

Cd. Universitaria, San Nicolás de los Garza, N.L., a 3 de noviembre de 2024

Participantes en la Actividad Fundamental #3

					
Datos del alumno	**Horacio Naranjo Rojas 2132048 ITS**	**Brian Orlando Ramírez Nuñez 2044753 IAS**	**Ángel Ricardo Álvarez García 2041139 ITS**	**Alejandra Garza Walle 2051321 ITS**	**Antonio Castillo Arreola 1952809 IAS**
					
Datos del alumno	**Alfonso Sainz Coronado 2049875 ITS**	**Emiliano Salinas Monsiváis 2055826 ITS**	**Valeria Guadalupe Martínez Tamez 2109508 IAS**	**Axel Arturo Loredo Pérez 2132062 ITS**	**Isac Alfredo Almaguer Espinosa 2049903 IAS**

índice

Introducción

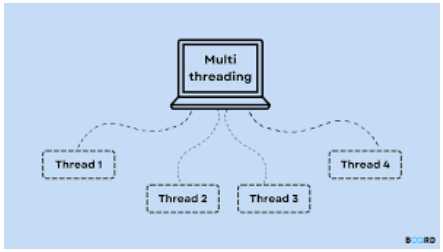
En el mundo actual, dominado por la informática y la tecnología, la capacidad de realizar múltiples tareas simultáneamente es crucial para el rendimiento y la eficiencia de los sistemas operativos. Este concepto, conocido como concurrencia, es fundamental para el funcionamiento de cualquier sistema operativo moderno, permitiendo que varias tareas o procesos se ejecuten de manera paralela o intercalada. La concurrencia no solo mejora el rendimiento del sistema, sino que también optimiza el uso de los recursos disponibles, como la CPU y la memoria, lo que es especialmente importante en un entorno en el que las aplicaciones y los usuarios exigen cada vez más.

Linux, uno de los sistemas operativos más robustos y versátiles, se ha destacado por su capacidad para manejar la concurrencia de manera eficiente y efectiva. Desde sus inicios, Linux ha sido diseñado para soportar múltiples procesos e hilos de ejecución, lo que lo convierte en una plataforma ideal para una amplia variedad de aplicaciones, desde servidores de alto rendimiento hasta dispositivos embebidos y sistemas de tiempo real. La concurrencia en Linux no solo se limita al nivel del usuario, donde los programas pueden crear y gestionar sus propios hilos y procesos, sino que también se gestiona a nivel del kernel, asegurando una asignación justa y eficiente de los recursos del sistema.

¿Qué es la concurrencia?

La concurrencia se puede lograr a través de varios mecanismos, siendo los más comunes el multitasking, el multithreading y el multiprocesamiento.

- **Multitasking:** Permite que un sistema operativo administre la ejecución de múltiples tareas (procesos) casi simultáneamente. El sistema asigna tiempo de CPU a cada proceso en intervalos cortos, de modo que parece que todos los procesos se están ejecutando al mismo tiempo.



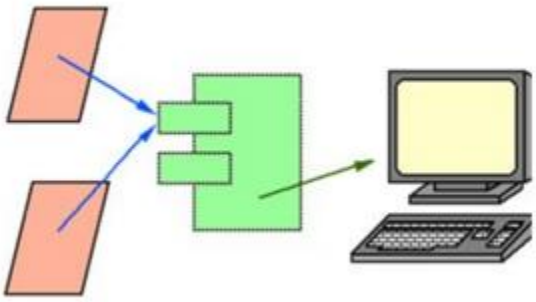
- **Multithreading:** Es una técnica que permite la ejecución concurrente de múltiples hilos dentro del mismo proceso. Los hilos comparten recursos, lo que permite una comunicación más eficiente y un uso optimizado del tiempo de CPU.

- **Multiprocesamiento**: Involucra la utilización de múltiples unidades de procesamiento (CPU) para ejecutar procesos concurrentes, lo que permite que el sistema operativo distribuya las tareas entre varios núcleos de CPU.

En esencia, la concurrencia es la gestión simultánea de múltiples tareas en un sistema. Esta capacidad es crucial para manejar la creciente demanda de aplicaciones que deben realizar diversas funciones al mismo tiempo, como reproducir música, descargar archivos y navegar por internet simultáneamente en un dispositivo.

Hay dos formas principales en las que la concurrencia se puede implementar en un sistema:

- **Paralelismo**: Es un tipo específico de concurrencia donde múltiples tareas se ejecutan realmente al mismo tiempo en diferentes núcleos de procesadores. Esto es posible en sistemas con múltiples procesadores o núcleos de CPU.



- **Multitarea:** En sistemas de un solo núcleo, el procesador ejecuta partes de diferentes tareas en intervalos cortos, cambiando rápidamente entre ellas. Esto da la apariencia de que las tareas se están ejecutando simultáneamente, aunque en realidad se están turnando para usar el procesador.

Tipos de Concurrencia

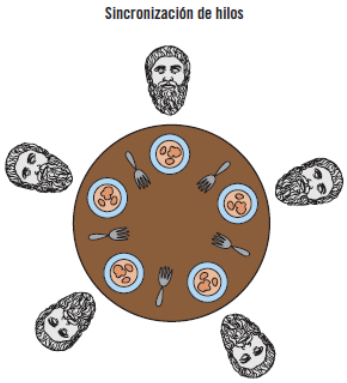
Concurrencia a Nivel de Proceso

La concurrencia a nivel de proceso es una de las formas más básicas de concurrencia y se refiere a la capacidad de un sistema operativo para gestionar la ejecución simultánea de múltiples procesos independientes. Cada proceso tiene su propio espacio de direcciones, lo que significa que no comparten memoria entre sí a menos que se implementen mecanismos de comunicación interproceso (IPC).

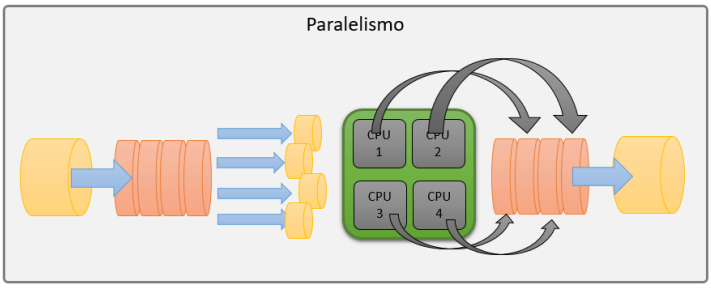
- **Multitarea Preemptiva:** En sistemas operativos modernos como Linux o Windows, la multitarea preemptiva es una técnica en la que el sistema operativo asigna tiempo de CPU a cada proceso y puede interrumpir un proceso en ejecución para dar paso a otro, asegurando que todos los procesos reciban una parte justa del tiempo de CPU.
- **Planificación de Procesos:** Para gestionar la concurrencia a nivel de proceso, los sistemas operativos utilizan algoritmos de planificación que determinan qué proceso debe ejecutarse en un momento dado. Los algoritmos más comunes incluyen la planificación de prioridad, la planificación de round-robin y la planificación de colas de múltiples niveles.

La concurrencia a nivel de hilo se refiere a la capacidad de un solo proceso para ejecutar múltiples hilos de manera concurrente. A diferencia de los procesos, los hilos dentro del mismo proceso comparten el mismo espacio de direcciones y pueden acceder a la misma memoria, lo que permite una comunicación y sincronización más eficiente.

- **Multithreading:** El multithreading es una técnica que permite la creación de múltiples hilos dentro de un proceso. Cada hilo puede ejecutar una parte diferente del código del programa al mismo tiempo. Esto es particularmente útil en aplicaciones que requieren realizar múltiples tareas simultáneamente, como el procesamiento de imágenes, la simulación o el manejo de múltiples conexiones de red.
- **Sincronización de Hilos:** Debido a que los hilos comparten memoria, es crucial que se implementen mecanismos de sincronización para evitar condiciones de carrera y otros problemas de consistencia. Primitivas como mutexes, semáforos y monitores son comúnmente utilizadas para garantizar que los hilos accedan a los recursos compartidos de manera segura.



Concurrencia a Nivel de Multiprocesador



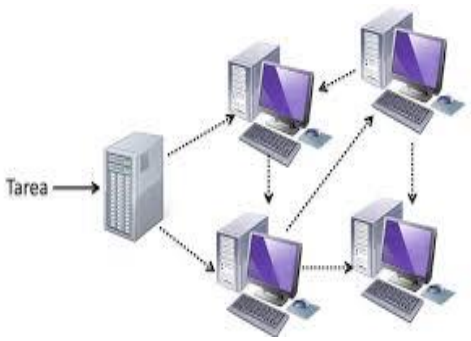
La concurrencia a nivel de multiprocesador, también conocida como paralelismo, se refiere a la capacidad de un sistema para utilizar múltiples unidades de procesamiento (CPU) para ejecutar diferentes procesos o hilos simultáneamente. Esto es particularmente relevante en sistemas con múltiples núcleos de CPU, donde la carga de trabajo se puede distribuir entre varios procesadores para mejorar el rendimiento.

- **Paralelismo de Datos:** En este enfoque, el mismo conjunto de instrucciones se aplica a múltiples datos en paralelo. Esto es común en aplicaciones científicas y de ingeniería donde grandes volúmenes de datos deben ser procesados simultáneamente.
- **Paralelismo de Tareas:** Aquí, diferentes tareas o funciones se ejecutan en paralelo en diferentes procesadores o núcleos. Este tipo de concurrencia es ideal para aplicaciones que pueden dividirse en sub-tareas independientes que no dependen entre sí.

Concurrencia Distribuida

La concurrencia distribuida se refiere a la ejecución concurrente de procesos o hilos en diferentes máquinas que están conectadas en red. Este tipo de concurrencia es fundamental en sistemas distribuidos, donde las tareas se dividen y ejecutan en múltiples nodos que pueden estar geográficamente dispersos.

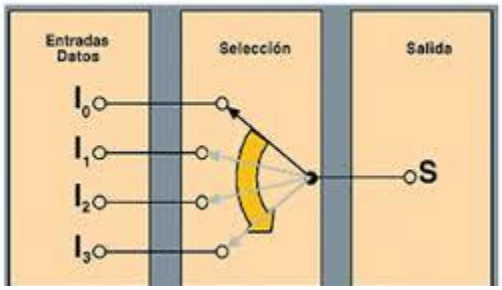
- **Computación Distribuida:** En la computación distribuida, las tareas se dividen en partes más pequeñas que se envían a diferentes nodos para ser ejecutadas en paralelo. Ejemplos incluyen sistemas de clústeres y entornos de computación en la nube, donde la concurrencia distribuida permite procesar grandes volúmenes de datos de manera eficiente.



- **Sincronización en Sistemas Distribuidos:** Sincronizar procesos y manejar la consistencia de los datos en un entorno distribuido es un desafío considerable. Protocolos como el consenso de Paxos y la consistencia eventual son utilizados para asegurar que los datos sean coherentes y que los procesos trabajen juntos sin conflictos.

Concurrencia a Nivel de Entrada/Salida (I/O)

La concurrencia a nivel de entrada/salida se refiere a la capacidad de un sistema para manejar múltiples operaciones de E/S de manera concurrente. Esto es crítico en sistemas que dependen en gran medida de la entrada y salida de datos, como servidores web, bases de datos y sistemas de archivos.



- **Operaciones de E/S Asíncronas:** En este modelo, las operaciones de E/S se realizan de manera asíncrona, permitiendo que otras tareas continúen ejecutándose mientras se espera que una operación de E/S termine. Esto mejora el rendimiento al reducir el tiempo de espera y optimizar el uso de la CPU.
- **Multiplexación de E/S:** Técnicas como la multiplexación de E/S permiten que un solo hilo maneje múltiples canales de entrada/salida simultáneamente, lo que es esencial para aplicaciones de red de alto rendimiento.

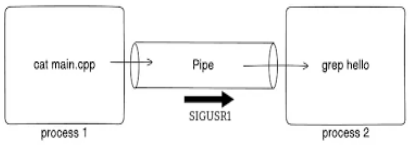
Modelos de programación concurrente

Memoria compartida:

- **Pthreads:** Linux ofrece la biblioteca POSIX Threads (pthreads), que permite crear y gestionar threads que comparten el mismo espacio de memoria.
- **Shared Memory (Memoria Compartida):** Linux proporciona mecanismos como shmget, shmat, shmdt, y shmctl para crear y gestionar segmentos de memoria compartida entre procesos.
- **Semáforos y Mutexes:** Para sincronizar el acceso a la memoria compartida, Linux proporciona semáforos.

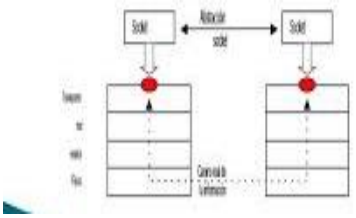
Paso de mensajes:

- **Pipes (Tuberías):** Los pipes son un mecanismo básico de comunicación entre procesos (IPC) en Linux que permite la transmisión de datos en un solo sentido.



- **Message Queues (Colas de Mensajes):** Linux ofrece colas de mensajes a través de las funciones msgget, msgsnd, msgrcv, y msgctl, que permiten la comunicación entre procesos mediante el envío y recepción de mensajes estructurados.

- **Sockets:** Los sockets son ampliamente utilizados para la comunicación entre procesos, tanto en la misma máquina como a través de una red. Permiten la comunicación bidireccional entre procesos.



Modelo de Actores:

- **Librerías y Frameworks:** Aunque Linux no implementa directamente un modelo de actores a nivel de sistema, existen librerías y frameworks que permiten su implementación, como Akka para Scala/Java o Erlang (el lenguaje Erlang fue diseñado específicamente para este modelo).

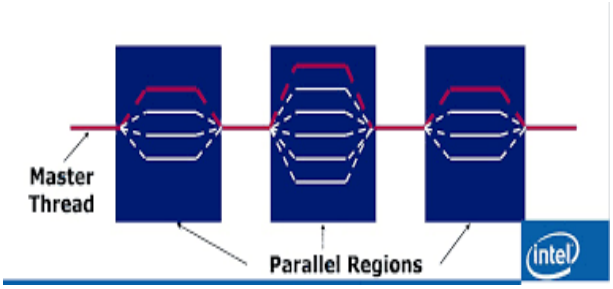
- **Goroutines y Canales en Go:** Aunque Go no es específico de Linux, su implementación en Linux permite utilizar goroutines y canales para emular un modelo de actores en la comunicación concurrente.

Modelos de Futuros y Promesas:

- **Librerías Asíncronas:** En lenguajes como C++s, Python (con asyncio), y Java (con CompletableFuture), se pueden implementar futuros y promesas en aplicaciones que se ejecutan en Linux.
- **Tareas Asíncronas en Linux:** Utilizando threads, hilos, y otras construcciones del sistema operativo, se pueden manejar tareas asíncronas que retornan resultados futuros.

Paralelismo de datos:

- **MPI y OpenMP:** En Linux, el paralelismo de datos se puede implementar utilizando MPI (Message Passing Interface) para la programación paralela en clústeres, o OpenMP para la paralelización en sistemas multicore.



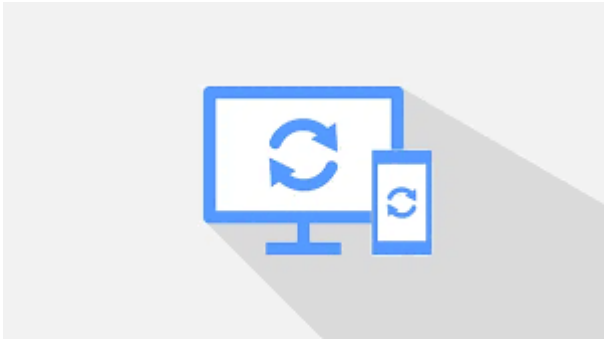
- **Herramientas de Procesamiento de Datos:** Linux soporta frameworks como Hadoop o Spark que implementan paralelismo de datos a gran escala.

La Concurrencia en Linux

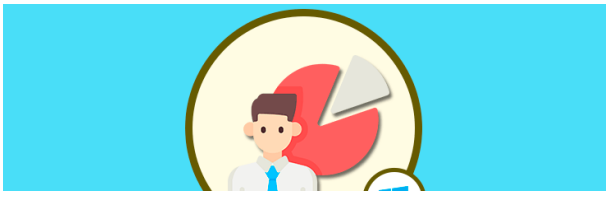
Linux es un sistema operativo que fue diseñado desde sus inicios para soportar la concurrencia de manera eficiente. Utiliza un modelo de multitarea con planificación por prioridades y es capaz de manejar múltiples procesos e hilos simultáneamente. Algunas de las características clave de la concurrencia en Linux incluyen:

- **Planificación de Procesos (Scheduling):** Linux utiliza un planificador de tareas que decide qué proceso o hilo debe ejecutarse en cada momento. A lo largo de los años, Linux ha implementado varios algoritmos de planificación, siendo uno de los más importantes el Completely Fair Scheduler (CFS). Este algoritmo garantiza que todos los procesos reciban una cantidad justa de tiempo de CPU, minimizando la latencia y asegurando una distribución equitativa de los recursos.

- **Sincronización:** La concurrencia trae consigo el desafío de la sincronización entre procesos e hilos. Linux proporciona varias primitivas de sincronización, como los mutexes, 12emáforos, y spinlocks, que aseguran que los recursos compartidos no sean modificados de manera inconsistente cuando son accedidos simultáneamente por múltiples hilos o procesos.



- **Espacio de Usuario y Espacio de Kernel:** En Linux, la concurrencia no solo se maneja en el espacio de usuario, donde los programas de los usuarios se ejecutan, sino también en el espacio del kernel, que es donde se realizan las operaciones de bajo nivel. Linux permite que las aplicaciones de usuario creen y gestionen sus propios hilos y procesos, mientras que el kernel maneja la asignación de recursos y la planificación global.



- **Manejo de Señales:** Las señales son una forma de interrumpir un proceso y que este tome una acción específica. En un entorno concurrente, las señales permiten que un proceso sea notificado de eventos como la finalización de otro proceso o la expiración de un temporizador. Linux maneja las señales de manera eficiente para asegurar que los procesos puedan responder a eventos de manera rápida y predecible.

Aplicaciones Prácticas de la Concurrencia en Linux

La concurrencia en Linux es utilizada en una amplia variedad de aplicaciones y entornos, desde servidores de alto rendimiento hasta sistemas embebidos. Por ejemplo, en un servidor web, la concurrencia permite que el servidor maneje múltiples solicitudes de usuarios simultáneamente, mejorando el tiempo de respuesta y la capacidad de manejo de la carga. En sistemas embebidos, la concurrencia permite que múltiples tareas, como la recolección de datos de sensores y la comunicación con otros dispositivos, se realicen en paralelo, maximizando la eficiencia del sistema.

Además, los desarrolladores de software pueden aprovechar las bibliotecas concurrentes en Linux, como POSIX threads (pthreads), para crear aplicaciones multihilo que ejecuten operaciones en paralelo, lo que es crucial en aplicaciones que requieren un alto rendimiento, como simulaciones, procesamiento de imágenes, y cálculos científicos.

Ejecución de concurrencia en el sistema operativo Linux

Primero que nada, recordemos que Linux es un sistema operativo open source que se compone del kernel, su elemento fundamental, y las herramientas, las aplicaciones y los servicios que se incluyen con él.

Para entender cómo es que Linux maneja la concurrencia, es buena idea voltear a ver a UNIX, aunque Linux no está basado directamente en Unix, ambos comparten muchas ideas y conceptos clave que son esenciales para entender cómo Linux maneja la concurrencia.

Entendemos por ejecución de concurrencia a la capacidad de un sistema operativo para ejecutar múltiples tareas simultáneamente o de manera superpuesta.

Ventajas de la ejecución de concurrencia en el sistema operativo Linux

Mejor utilización de recursos

La concurrencia permite una mejor utilización de los recursos del sistema, como CPU, memoria y dispositivos de E/S. Al permitir que se ejecuten múltiples tareas simultáneamente, el sistema puede hacer un mejor uso de sus recursos disponibles.

Linux es conocido por su eficiencia en la gestión de recursos, especialmente en entornos de servidor y sistemas embebidos. Por ejemplo, en un servidor Linux, el kernel puede priorizar tareas críticas mientras relega otras a segundo plano, utilizando la programación de tareas. Esto asegura que la CPU esté ocupada ejecutando tareas útiles, sin desperdiciar ciclos de reloj.

Mejor capacidad de respuesta

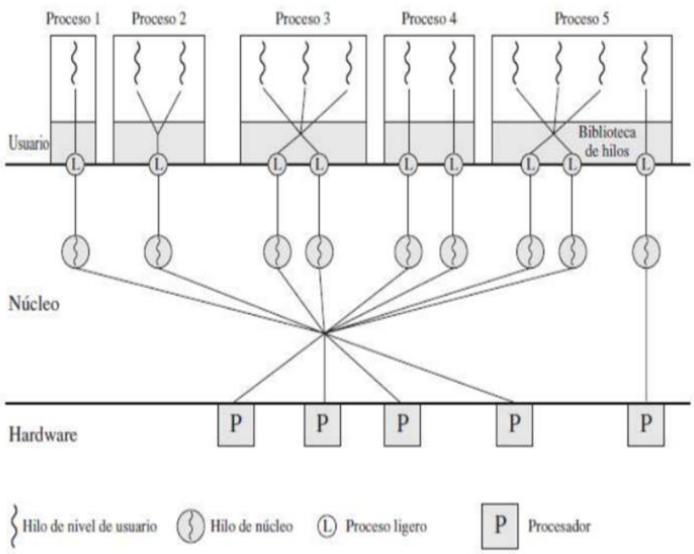
La simultaneidad puede mejorar la capacidad de respuesta del sistema al permitir que se ejecuten múltiples tareas simultáneamente.

Esto es particularmente importante en sistemas en tiempo real y aplicaciones interactivas, como juegos y multimedia.

Mayor velocidad y eficiencia

La concurrencia permite ejecutar múltiples tareas simultáneamente, mejorando el rendimiento general del sistema. Al utilizar múltiples procesadores o subprocesos, las tareas se pueden ejecutar en paralelo, lo que reduce el tiempo total de procesamiento.

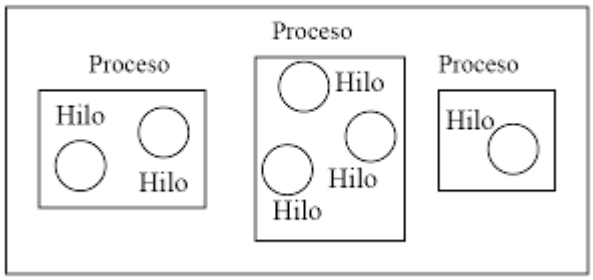
Linux aprovecha el multiprocesamiento simétrico (SMP) para ejecutar tareas en paralelo en sistemas con múltiples núcleos. Un ejemplo claro es el uso de Linux en supercomputadoras, como en los sistemas de la lista TOP500, donde se aprovechan miles de núcleos para realizar cálculos científicos complejos en paralelo. Además, la función Cgroups, permite a Linux asignar recursos de manera granular a diferentes tareas, optimizando el rendimiento general del sistema.



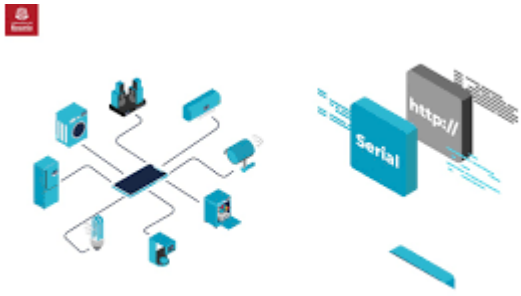
Facilitación del desarrollo de aplicaciones distribuidas

Una de las grandes ventajas de la concurrencia en Linux es cómo facilita el desarrollo de aplicaciones distribuidas. Las aplicaciones distribuidas son aquellas que se ejecutan en múltiples sistemas conectados en red, lo que permite dividir el procesamiento y mejorar la eficiencia.

- **Soporte para Múltiples Procesos e Hilos:** Linux, con su arquitectura concurrente, permite que los desarrolladores creen aplicaciones que pueden ejecutarse en múltiples procesos e hilos. Esto es crucial en entornos distribuidos, donde diferentes componentes de una aplicación pueden estar corriendo en distintos nodos de una red. La capacidad de Linux para manejar procesos e hilos de manera eficiente asegura que las aplicaciones distribuidas puedan escalar y funcionar correctamente.



- **Comunicación Interprocesos (IPC):** Linux proporciona una amplia gama de mecanismos de comunicación entre procesos (IPC) como tuberías, colas de mensajes, memoria compartida y semáforos. Estos mecanismos son fundamentales en el desarrollo de aplicaciones distribuidas, ya que permiten la sincronización y el intercambio de datos entre procesos que pueden estar ejecutándose en diferentes máquinas dentro de la red.
- **Redes y Protocolos:** Linux tiene un fuerte soporte para redes y protocolos, facilitando la creación de aplicaciones distribuidas que requieren comunicación constante entre diferentes sistemas. Esto incluye el soporte para sockets, que son esenciales para establecer conexiones de red entre procesos en diferentes máquinas.



Escalabilidad

La concurrencia puede mejorar la escalabilidad del sistema al permitirle manejar un número cada vez mayor de tareas y usuarios sin degradar el rendimiento.

Linux gracias a su escalabilidad, es utilizado en infraestructuras grandes como servidores web, lo que demuestra su capacidad para manejar grandes volúmenes de tareas y usuarios simultáneamente sin degradación del rendimiento.

Robustez y tolerancia a fallos

La concurrencia puede mejorar la tolerancia a fallas del sistema al permitir que las tareas se ejecuten de forma independiente. Si una tarea falla, no afecta la ejecución de otras tareas.

Si un proceso falla, Linux puede aislar el fallo utilizando técnicas como cgroups y namespaces, asegurando que el resto del sistema continúe funcionando sin interrupciones.

Estados de un proceso

Mientras un proceso se ejecuta su estado cambia debido a sus circunstancias:

Running: el proceso se está ejecutando o está listo para ello (esperando a que se le asigne tiempo de CPU)

Waiting: el proceso está esperando un evento o un recurso. Aquí Linux diferencia entre 2 tipos, interrumpible e ininterrumpible. Mientras el primero puede detenerse mediante una señal los ininterrumpibles están esperando a una condición del hardware no puede ser detenido bajo ninguna circunstancia.

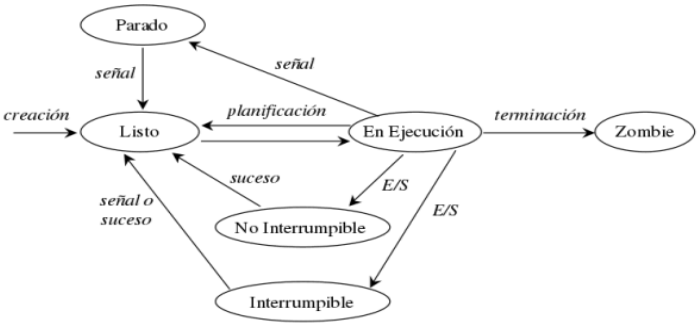
Procesos Hijos (o Subprocesos): En Linux, los procesos pueden crear otros procesos mediante la llamada al sistema `fork()`. El proceso que hace la llamada se denomina "proceso padre" y el proceso creado es el "proceso hijo". Ambos procesos pueden ejecutarse concurrentemente y el hijo puede ejecutar una copia del programa del padre o cargar un programa completamente diferente usando `exec()`.

Procesos Hilos (Threads): Son una forma de procesos ligeros que comparten el mismo espacio de memoria y recursos del proceso padre. Los hilos permiten que diferentes partes de un programa se ejecuten de forma concurrente en un solo proceso, aprovechando al máximo las capacidades de los procesadores multinúcleo

Procesos de Tiempo Real: Son procesos que deben cumplir con restricciones de tiempo específicas para funcionar correctamente. Linux proporciona mecanismos para crear y gestionar procesos de tiempo real, que son esenciales para aplicaciones sensibles al tiempo, como sistemas embebidos o aplicaciones de audio y video en tiempo real.

Stopped: el proceso está detenido, usualmente debido a recibir una señal.

Zombie: es un proceso que ha completado su ejecución pero aún tiene una entrada en la tabla de procesos, permitiendo al proceso que lo ha creado leer el estado de su salida.



Estos tipos de procesos concurrentes permiten a Linux gestionar múltiples tareas simultáneamente, optimizando el uso de recursos del sistema y mejorando la eficiencia general del sistema operativo.

Interacciones en la concurrencia de Linux

Los procesos concurrentes en Linux pueden interactuar de diversas formas.

La coordinación y comunicación entre procesos son claves para el rendimiento y la estabilidad del sistema.

Comunicación entre Procesos (IPC)

Tuberías (Pipes): Este tipo de comunicacion es unidireccional entre procesos y la salida de un proceso es la entrada de otro

Tuberías con nombre (FIFOs): Utilizan comunicacion bidireccional y son usadas por procesos sin relacion padre-hijo

Colas de mensajes: Envio y recibo de mensajes en forma de cola

Memoria Compartida: Multiples procesos que acceden al mismo segmento de memoria. Es un metodo rapido para intercambiar datos.

Sincronización entre Procesos

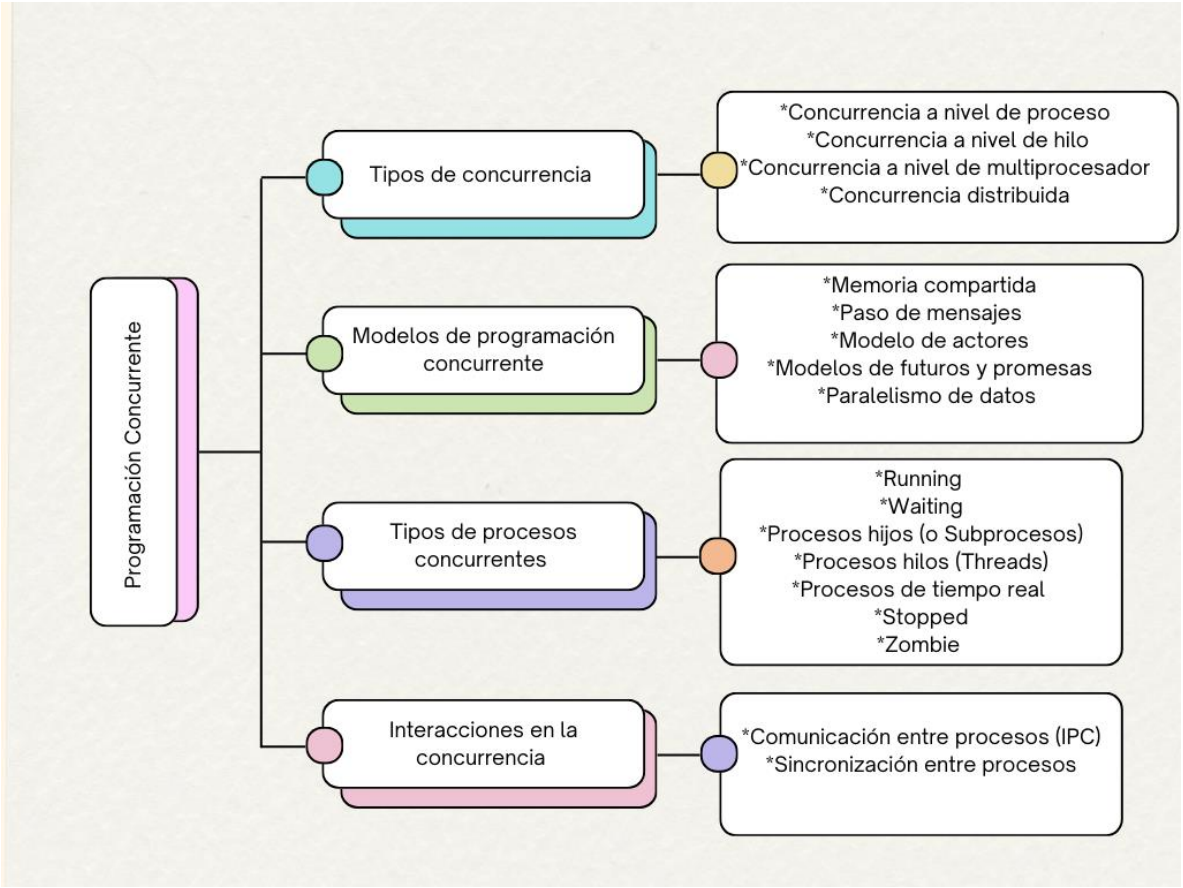
Locks (Mutexes y Spinlocks): Aseguran que solo un proceso acceda a un recurso crítico a la vez.

Condiciones de Variables: Son procesos que suspenden la ejecución hasta que una condición sea verdadera.

Semáforos: Controlan el acceso a recursos compartidos y evitan condiciones de carrera.

Banderas de Eventos: Facilitan la señalización de eventos entre procesos.

Cuadro Sinoptico



Conclusión General

La concurrencia es un pilar fundamental en la arquitectura de sistemas operativos y en el desarrollo de aplicaciones modernas, dado que permite la ejecución simultánea de múltiples tareas, optimizando así el uso de los recursos del sistema y mejorando el rendimiento general. A lo largo de este ensayo, hemos explorado los diferentes tipos de concurrencia, cada uno con sus propias características, aplicaciones y desafíos.

La concurrencia a nivel de proceso nos permite manejar la ejecución de múltiples programas independientes, aprovechando la capacidad del sistema operativo para distribuir el tiempo de CPU de manera equitativa entre ellos. Esto es esencial en cualquier entorno multitarea, donde múltiples aplicaciones deben ejecutarse sin interferir unas con otras. Por su parte, la concurrencia a nivel de hilo introduce un nivel adicional de granularidad, permitiendo que diferentes partes de un mismo programa se ejecuten simultáneamente. Esto no solo mejora la eficiencia de las aplicaciones, sino que también facilita la creación de software más reactivo y escalable, capaz de manejar múltiples operaciones de manera fluida.

Conclusiones Individuales

Antonio Castillo Arreola 1952809 IAS

La concurrencia en los sistemas operativos que permite la ejecución simultánea de múltiples procesos o hilos dentro de un sistema. La importancia de la concurrencia radica en su capacidad para mejorar la eficiencia y la velocidad de los sistemas multitarea, especialmente en entornos con múltiples usuarios o aplicaciones que requieren procesamiento simultáneo. También es esencial en sistemas en tiempo real donde las tareas deben completarse dentro de límites de tiempo estrictos.

En resumen, la concurrencia es vital para los sistemas operativos modernos porque mejora la eficiencia y la capacidad de respuesta, pero requiere técnicas cuidadosas de gestión y control para garantizar el funcionamiento seguro y efectivo del sistema.

Angel Ricardo Alvarez Garcia 2041139 ITS

En nuestro trabajo se investigó acerca de los temas fundamentales en las que un sistema operativo funciona, por ejemplo, de las múltiples tareas, el cómo optimizan los recursos de software, la importancia de trabajar del sistema operativo que escogimos (Linux), el cómo los procesos cambian de estado, la capacidad que tiene el sistema operativo usa múltiples procesos para tener mejor rendimiento dentro del sistema, etc. Yo en concreto me enfoqué más en investigar los modelos de concurrencia, se investigó que hay varios modelos y que cada uno tiene diferentes características para las diferentes aplicaciones, en conclusión, todos estos temas son totalmente fundamentales para que nosotros podamos entender el funcionamiento de los diferentes sistemas operativos.

Valeria Guadalupe Martínez Tamez 2109508 IAS

En conclusión, entendí que la concurrencia se refiere a un tipo de proceso que se observa en diferentes áreas, un poco más en programación, en donde los programas ejecutan varios procesos simultáneamente. Se puede decir que la

concurrencia ocurre constantemente, permitiendo que varios procesos se lleven a cabo al mismo tiempo, y cada uno de estos procesos interactúa y contribuye de manera específica.

Horacio Naranjo Rojas 2132048 ITS

El multiprocesamiento simétrico (SMP, por sus siglas en inglés) es una arquitectura de procesamiento en la que múltiples procesadores comparten la misma memoria y el mismo conjunto de recursos de entrada y salida, operando bajo un solo sistema operativo. Cada procesador en un sistema SMP tiene acceso equitativo a los recursos del sistema, lo que permite una distribución más equilibrada de las cargas de trabajo y una mejora en la eficiencia del procesamiento paralelo. La principal ventaja de SMP es su capacidad para mejorar el rendimiento y la escalabilidad del sistema al permitir que múltiples tareas se realicen simultáneamente, lo que resulta en un mejor aprovechamiento del hardware y una mayor capacidad de respuesta en aplicaciones que requieren alta capacidad de procesamiento. Sin embargo, la complejidad de la gestión de recursos compartidos y la sincronización entre procesadores puede ser un desafío, especialmente en sistemas con un gran número de procesadores.

Alfonso Sainz Coronado 2049875 ITS

Dado todo lo que investigamos, concluyó que la concurrencia es de gran utilidad a la hora de mejorar la eficiencia en los sistemas operativos, dado que nos permite ejecutar múltiples procesos simultaneamente, ésto es muy importante a la hora de realizar proyectos grandes los cuales requieran de una gran cantidad de procesos que ejecutar.

Emiliano Salinas Monsiváis 2055826

La concurrencia en Linux es un aspecto fundamental que permite al sistema operativo gestionar múltiples tareas de manera eficiente y simultánea. A través de técnicas como multitarea, hilos y la coordinación entre procesos, Linux maximiza el uso de los recursos del sistema, garantizando que las aplicaciones se ejecuten sin conflictos y de forma eficiente. Este manejo concurrente es esencial para el rendimiento en entornos de servidores y equipos de escritorio, permitiendo a Linux ofrecer una plataforma buena y confiable para una variedad de aplicaciones.

Isac Alfredo Almaguer Espinosa, 2049903, IAS

En esta investigación aprendí sobre la concurrencia en Linux que se trata de cómo el sistema maneja varias tareas a la vez, eso significa que el sistema es muy bueno en dividir su tiempo entre diferentes tareas de manera que parece que todo ocurre simultáneamente.

También aprendí sobre los hilos, que son como mini-programas que pueden correr al mismo tiempo dentro de un proceso más grande, así como de SMP especialmente que es la multiprocesación simetríca y como ejecuta estos hilos de manera eficiente, repartiendo la carga entre todos ellos.

La concurrencia es un concepto más amplio que abarca cómo Linux gestiona múltiples tareas, y los hilos y SMP son herramientas o mecanismos que el sistema utiliza para llevar a cabo esa concurrencia de manera eficiente.

Axel Arturo Loredo Pérez 2132062 ITS

A partir de esta investigación, aprendimos cómo los sistemas operativos optimizan la ejecución y gestión de tareas. Se entiende que los procesos y hilos son fundamentales para la ejecución eficiente, con los hilos permitiendo mayor paralelismo. Además, el SMP permite a múltiples procesadores trabajar simultáneamente, mejorando el rendimiento. También que los micro núcleos simplifican la gestión del sistema al tener solo funciones esenciales en el núcleo.

Podemos concluir que Linux es eficaz en la gestión de hilos en entornos SMP, maximizando la eficiencia y el rendimiento del sistema.

Brian Orlando Ramirez Nuñez 2044753 IAS

Bueno ya para concluir puedo enfocarme en decir que la concurrencia de sistemas operativos es algo sumamente necesario puesto que optimiza demasiado las cosas y hace que todo sea muchísimo más rápido, esto pues gracias a una gestión increíble de los núcleos e hilos que pueda hacer uso el SO.

Todo lo que hemos investigado muestra que la concurrencia no solo optimiza el rendimiento, sino que también abre la puerta a sistemas más robustos y capaces de manejar múltiples tareas al mismo tiempo, mejorando la eficiencia y la capacidad de respuesta de nuestras aplicaciones

Alejandra Garza Walle 2051321 ITS

Una vez contribuido en la investigación, llegado a la conclusion concurrencia es un concepto central en la informática, que se refiere a la capacidad de ejecutar múltiples tareas o procesos de manera simultánea o intercalada, optimizando el uso de los recursos del sistema y mejorando la eficiencia global. Cada uno diseñado para diferentes contextos de operación: a nivel de proceso, a nivel de hilo, en sistemas multiprocesador, distribuidos y en operaciones de entrada/salida. Cada tipo permite gestionar y aprovechar al máximo los recursos disponibles, ya sea a través de la ejecución paralela de procesos en múltiples núcleos de CPU o mediante la gestión eficiente de hilos dentro de un mismo proceso.

La ejecución concurrente ofrece numerosas ventajas, como la mejora en la capacidad de respuesta de las aplicaciones, la maximización de la eficiencia de los recursos y la posibilidad de realizar múltiples tareas de forma simultánea, lo que es especialmente beneficioso en aplicaciones que requieren alta disponibilidad o en entornos de red complejos.

Referencias bibliográficas

Definición de Linux. (2016, 2 Abril). https://www.redhat.com/es/topics/linux/what-is-linux#:~:text=Art%C3%ADculo-,%C2%BFQu%C3%A9%20es%20Linux%3F,que%20se%20incluyen%20con%20%C3%A9l.

¿Cuáles son los principales beneficios y desafíos de la concurrencia en los sistemas operativos? (2023, 1 septiembre). www.linkedin.com. https://es.linkedin.com/advice/0/what-main-benefits-challenges-concurrency-operating?lang=es#:~:text=La%20concurrencia%20es%20la%20capacidad,la%20escalabilidad%20y%20el%20paralelismo.

Alvarez, B. C. (2018, 18 abril). Procesos en Linux. UNALMED. https://so-g8-linux-ubuntu.blogspot.com/2018/03/concurrencia-en-ubuntu.html?m=1

Concurrencia en el sistema operativo. (2022, 04 septiembre). https://es.linux-console.net/?p=23391#google_vignette

La programación concurrente y su utilidad actual y futura. (2023, 23 marzo). Unir. Recuperado 31 de agosto de 2024, de https://www.unir.net/ingenieria/revista/programacion-concurrente/

Los procesos UNIX — Programación concurrente en Linux. (s. f.). https://ocw.ehu.eus/pluginfile.php/48902/mod_resource/content/13/html/Recursos/P03/Resumen_procesos.html

Deana. (2023b, agosto 18). *Concurrencia en la programación*. TechEdu. https://techlib.net/techedu/concurrencia-en-la-programacion/

Salguero, E. (2019, 12 octubre). Sistemas distribuidos: transacciones y control de concurrencia (III). *Medium*. https://medium.com/@edusalguero/sistemas-distribuidos-transacciones-y-control-de-concurrencia-1f5845edd5f

Bovet, D. P., & Cesati, M. (2005). Understanding the Linux Kernel. O'Reilly Media.

Love, R. (2010). Linux Kernel Development. Addison-Wesley Professional.

Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts. Wiley.

Caballero, D. (2024, 22 enero). Qué es Linux: el sistema operativo de código abierto. ADSLZone. https://r.search.yahoo.com/_ylt=AwrNaAuAW9FmFjgU5g_D8Qt.;_ylu=Y29sbwNiZjEEcG9zAzQEdnRpZAMEc2VjA3Ny/RV=2/RE=1725025280/RO=10/RU=https%3a%2f%2fwww.adslzone.net%2freportajes%2fsoftware%2fque-es-linux%2f/RK=2/RS=MHJGuy7pkfakQmCg2lyHQSrqJVA-

Recalde, M. F. (2020, 11 agosto). Qué es Linux y Cómo Funciona: Guía Fácil y Completa 2024. Tu Programa Para. https://tuprogramapara.com/linux/

Procesos e hilos — Programación concurrente en Linux. . https://ocw.ehu.eus/pluginfile.php/48902/mod_resource/content/13/html/Recursos/P04/Procesos_y_threads.html

Gestión de procesos — Programación concurrente en Linux. . https://ocw.ehu.eus/pluginfile.php/48902/mod_resource/content/13/html/Recursos/P03/Gestion_procesos.html

Campos, O. (2011, 12 septiembre). *Programando módulos para el Kernel de Linux. Concurrencia en el Kernel*. Genbeta. https://www.genbeta.com/desarrollo/programando-modulos-para-el-kernel-de-linux-concurrencia-en-el-kernel